HY-DRO-GEN

User guide

v0.1.2 2025-09-28 MIT

Word hyphenation via bindings to typst/hypher

Neven Villani
✉ neven@crans.org

HY-DRO-GEN can split words into syllables in any supported language, which enables correct hyphenation.

HY-DRO-GEN is composed of
1. an internal WASM module that provides bindings to the hyphenation library natively used for Typst, typst/hypher,
2. a public layer to abstract away the internal details.

This manual is only concerned with the latter.

Contributions

If you have ideas for improvements, or if you encounter a bug, you are encouraged to contribute to HY-DRO-GEN by submitting a bug report, feature request, or pull request.

Versions

- `dev`
- `hy-dro-gen:0.1.2` (`latest`) → `hypher:0.1.6` forked to `606d415`
- `hy-dro-gen:0.1.1` → `hypher:0.1.6`
- `hy-dro-gen:0.1.0` → `hypher:0.1.5`

Table of Contents

Quick start **4**

Language validation **5**

II.1 Existence check 5

II.2 Fallback 6

Dynamically loaded languages **7**

III.1 Some background 7

III.2 Obtaining tries 7

III.2.1 Download pattern files 7

III.2.2 Install `hypher` 7

III.2.3 Compile the trie 8

III.3 Loading patterns 8

III.3.1 Manual 8

III.3.2 Automatic 8

API **9**

About **11**

V.1 Useful resources 11

V.2 Dependencies 11

Part I

Quick start

Import the latest version of HY-DRO-GEN with:

```
#import "@preview/hy-dro-gen:0.1.2" as hy
```

The main function provided by HY-DRO-GEN is `#hy.syllables`, which takes as input a word and a language (specified by its ISO 639-1 code), and returns the word split by syllables. By default, the language is `"en"`, i.e. English.

`#hy.syllables("hydrogen")`	`("hy", "dro", "gen")`
`#hy.syllables("hydrogène", lang: "fr")`	`("hy", "dro", "gène")`
`#hy.syllables("υδρογόνο", lang: "el")`	`("υ", "δρο", "γό", "νο")`

Part II

Language validation

If a language is unsupported, the default behavior is a panic.

`#hy.syllables("hydrogène", lang: "xz")`	**panic: Invalid language**

In the eventuality that you need to hyphenate for an arbitrary language that is not guaranteed to be a valid ISO 639-1 code, it is recommend that you either validate the language or specify a fallback.

II.1 Existence check

The function `#hy.exists` checks if a language is natively supported. If `#hy.exists` returns `true`, a call of `#hy.syllables` is guaranteed to not panic given this language. Since all ISO 639-1 codes have two letters, any string of more than two letters given to this function will always produce `false`.

`#hy.exists("en")`	`true`
`#hy.exists("xz")`	`false`
`#hy.exists("foobar")`	`false`

Here is the list of all languages supported natively:

Code	Language		Code	Language		Code	Language
`"af"`	Afrikaans		`"sq"`	Albanian		`"be"`	Belarusian
`"bg"`	Bulgarian		`"ca"`	Catalan		`"hr"`	Croatian
`"cs"`	Czech		`"da"`	Danish		`"nl"`	Dutch
`"en"`	English		`"et"`	Estonian		`"fi"`	Finnish
`"fr"`	French		`"ka"`	Georgian		`"de"`	German
`"el"`	Greek		`"hu"`	Hungarian		`"is"`	Icelandic
`"it"`	Italian		`"ku"`	Kurmanji		`"la"`	Latin
`"lt"`	Lithuanian		`"mn"`	Mongolian		`"no"`	Norwegian
`"nb"`	Norwegian		`"nn"`	Norwegian		`"pl"`	Polish
`"pt"`	Portuguese		`"ru"`	Russian		`"sr"`	Serbian
`"sk"`	Slovak		`"sl"`	Slovenian		`"es"`	Spanish
`"sv"`	Swedish		`"tr"`	Turkish		`"tk"`	Turkmen
`"uk"`	Ukrainian						

The same list available via the static dictionary `#hy.languages`.

II.2 Fallback

Alternatively, you can provide a fallback strategy among:

- `auto`: languages that do not exist will silently be skipped,
- `str`: a valid ISO 639-1 code as a fallback will be used in the event that ⟨`lang`⟩ is invalid.

`#hy.syllables("hydrogène", lang: "xz")`	**panic: Invalid language**
`#hy.syllables("hydrogène", lang: "xz", fallback: auto)`	`("hydrogène",)`
`#hy.syllables("hydrogène", lang: "xz", fallback: "fr")`	`("hy", "dro", "gène")`

Part III

Dynamically loaded languages

This feature is experimental and lacks validation. If you do not follow the instructions below you can end up with incomprehensible error messages.

III.1 Some background

As explained in the original blog post for hypher, hyphenation in Typst works by generating an automaton from a TEX pattern file. In practice this is implemented by the crate `hypher`. By default `hypher`, and thus Typst, embeds the automata for 35 (possibly soon 36) languages, but until issue #5223 lands, it is not currently possible to load custom patterns.

III.2 Obtaining tries

III.2.1 Download pattern files

There are a number of hyphenation pattern files available on hyphenation.org, of which quite a few are not available natively in Typst.

In what follows I assume that you have downloaded your pattern files, and saved them to `patterns/hyph-${iso}.tex`, replacing `${iso}` with whatever code the language you want to use has. Create also a directory `tries/`.

III.2.2 Install `hypher`

This step is still very rough. It'll get better once some of my local changes have been upstreamed to typst/hypher.

The `.tex` pattern files need to be compiled to automata readable by hypher. First we need to install hypher locally as a binary. Currently the only way I know of doing so is:

```
# Download the fork of hypher that can compile tries
$ cd /tmp && git clone https://github.com/Vanille-N/hypher.git
# Install it locally
$ cargo install --path hypher --features bin
# Go back to your workspace and check that it works.
$ cd - && hypher --help
```

I hope that soon this process can be simplified to:

```
$ cargo install hypher --features bin
```

III.2.3 Compile the trie

With `hypher` now installed, run

```
$ hypher build patterns/hyph-${iso}.tex tries/${iso}.bin
```

III.3 Loading patterns

`#hy.syllables` can take as `⟨lang⟩` an automata as bytes.

```
#hy.syllables(
  "galego",
  lang: read("tries/gl.bin", encoding: none),
)
```

```
("ga", "le", "go")
```

III.3.1 Manual

If you want to hyphenate a specific piece of text with a pattern, you can write for example:

```
#let trie = read("tries/gl.bin", encoding: none)
#show regex(\w+): word => {
  syllables(word.text, lang: trie).join([-?])
}
#text(lang: "gl")[#my-text]
```

III.3.2 Automatic

Altertatively, you can use `#hy.load-patterns` and `#hy.apply-patterns`. Behind the scenes they will perform almost the same manipulation as in Section III.3.1.

```
#hy.load-patterns(
  gl: read("tries/gl.bin", encoding: none),
  // accepts multiple pairs in the format
  // {iso}: read("tries/{iso}.bin", encoding: none),
)
#show: hy.apply-patterns("gl")
#text(lang: "gl")[#my-text]
```

Part IV
API

```
#hy.apply-patterns          #hy.load-patterns
#hy.exists                  #hy.syllables
```

`#hy.apply-patterns(⟨iso⟩) → function`

↑ Since 0.1.2

Apply show rules to hyphenate the specified language. The output is a `(content)→content` that can be used as `#show` rule for the rest of the document.

Argument

`⟨iso⟩` `iso`

ISO 639-1 code of a language previously added by `#hy.load-patterns`.

`#hy.exists(⟨iso⟩) → bool`

Check if a code corresponds to a language that has registered patterns.

See the list of officially supported languages at `github:typst/hypher`

If this function returns `true`, then an invocation of `#hy.syllables` with this language is guaranteed to not raise an "Invalid language" failure.

Argument

`⟨iso⟩` `iso`

2-letter ISO 639-1 code, e.g `"en"`, `"fr"`, `"el"`, etc.

`#hy.load-patterns(..⟨args⟩) → content`

↑ Since 0.1.2

Load new precompiled patterns. If your patterns are not compiled yet, see Section III.2.2 and Section III.2.3.

Argument

`..⟨args⟩` `dictionary`

One or more pairs in the format `{iso}: {bytes}`, for example one could write:

```
#load-patterns(
  en: read("tries/en.bin", encoding: none),
  fr: read("tries/fr.bin", encoding: none),
)
```

`#hy.syllables(⟨word⟩, ⟨lang⟩: "en", ⟨fallback⟩: none, ⟨dyn⟩: false) → (..string,)`

Splits a word into syllables according to available hyphenation patterns.

Argument

⟨word⟩ `str`

Word to split.

Argument

⟨lang⟩: "en" `iso` | `bytes`

Either an ISO 639-1 code, or bytes representing a trie.

Argument

⟨fallback⟩: none `none` | `auto` | `iso`

Determines the behavior in case `lang` is unsupported

- `none`: panics with "Invalid language"
- `auto`: the word is not split at all
- `iso`: use that instead

Argument

⟨dyn⟩: false `bool`

↑ Since 0.1.2

context

Look also in the dynamically loaded languages, i.e. valid values for ⟨lang⟩ now include not just the builtin ones but also those declared via `#hy.load-patterns`. Setting this to true will also make the function contextual,

#languages `dictionary`

Dictionary of supported codes and languages, in the format:

```
(en: "English", fr: "French", ...)
```

This dictionary is expected but not guaranteed to be in sync with `exists`, because they are fetched through different means. (`exists` queries the actual WASM module, while `languages` is generated automatically from the source code of `hypher`). If they are out of sync, `exists` is the authority for which languages are actually supported by `syllables`.

Part V
About

V.1 Useful resources

- How to put 30 Languages Into 1.1MB is the blog post that introduced typst/hypher,
- https://www.hyphenation.org/ is a repository of hyphenation patterns.

V.2 Dependencies

HY-DRO-GEN is obviously dependent on typst/hypher its main dependency. Currently it actually uses a fork Vanille-N/hypher, since dynamically loading tries is not supported by typst/hypher, but I am open to upstreaming all the features that the Typst project finds desirable.

This manual is built with MANTYS and TIDY.